# 《药店销售管理平台概要设计文档》

## 目录

1. 引言 1.1 目的 1.2 项目背景 1.3 术语和缩写
2. 架构设计 2.1 系统架构概述 2.2 技术选型 2.3 模块划分
3. 数据库设计 3.1 数据库概述 3.2 数据库表设计
4. 后端设计 4.1 项目结构 4.2 核心功能设计 4.3 API接口设计
5. 前端设计 5.1 项目结构 5.2 核心功能设计 5.3 页面设计
6. 安全设计 6.1 身份验证与授权 6.2 数据加密 6.3 日志与监控
7. 部署设计 7.1 部署环境 7.2 部署流程

### 1. 引言

#### 1.1 目的

本文档描述了药店销售管理平台的概要设计，包括系统架构、模块设计、数据库设计、前后端设计、安全设计以及部署设计。

#### 1.2 项目背景

为了提高药店的运营效率和管理水平，开发一套基于Spring Boot和Vue的药店销售管理平台，方便药店进行药品、销售、库存、供应商等业务的管理。

#### 1.3 术语和缩写

* Spring Boot：一种基于Java的开源框架，用于简化Spring应用的开发。
* Vue：一种渐进式JavaScript框架，用于构建用户界面。
* POS：销售点系统（Point of Sale）。

### 2. 架构设计

#### 2.1 系统架构概述

系统采用前后端分离的架构，前端使用Vue框架，后端使用Spring Boot框架，前后端通过RESTful API进行数据交互。系统架构图如下：

+----------------+ +----------------+  
| 前端(Vue) | <----> | 后端(Spring) |  
+----------------+ +----------------+  
 ^ |  
 | v  
 +-----------+ +-----------+  
 | 浏览器 | | 数据库 |  
 +-----------+ +-----------+

#### 2.2 技术选型

* 后端：Spring Boot
* 前端：Vue
* 数据库：MySQL
* API协议：RESTful
* 安全：Spring Security, JWT

#### 2.3 模块划分

系统主要分为以下几个模块： - 用户管理模块 - 药品管理模块 - 销售管理模块 - 库存管理模块 - 供应商管理模块 - 报告与分析模块 - 系统设置模块

### 3. 数据库设计

#### 3.1 数据库概述

数据库采用MySQL，设计遵循第三范式，确保数据的一致性和完整性。

#### 3.2 数据库表设计

* 用户表（users）
  + id: 主键
  + username: 用户名
  + password: 密码
  + role: 角色
  + created\_at: 创建时间
  + updated\_at: 更新时间
* 药品表（medicines）
  + id: 主键
  + name: 药品名称
  + category: 药品类别
  + price: 药品价格
  + description: 药品描述
  + created\_at: 创建时间
  + updated\_at: 更新时间
* 销售记录表（sales）
  + id: 主键
  + medicine\_id: 药品ID（外键）
  + quantity: 销售数量
  + total\_price: 总价
  + sale\_date: 销售日期
  + created\_at: 创建时间
  + updated\_at: 更新时间
* 库存表（inventory）
  + id: 主键
  + medicine\_id: 药品ID（外键）
  + quantity: 库存数量
  + batch\_number: 批次号
  + expiration\_date: 有效期
  + created\_at: 创建时间
  + updated\_at: 更新时间
* 供应商表（suppliers）
  + id: 主键
  + name: 供应商名称
  + contact\_info: 联系信息
  + created\_at: 创建时间
  + updated\_at: 更新时间

### 4. 后端设计

#### 4.1 项目结构

后端项目结构如下：

pharmacy-management-platform/  
├── src/  
│ ├── main/  
│ │ ├── java/  
│ │ │ └── com/  
│ │ │ └── example/  
│ │ │ └── pharmacymanagement/  
│ │ │ ├── controller/  
│ │ │ ├── service/  
│ │ │ ├── repository/  
│ │ │ ├── model/  
│ │ │ └── config/  
│ │ └── resources/  
│ │ ├── application.properties  
│ │ └── static/  
│ └── test/  
├── pom.xml  
└── README.md

#### 4.2 核心功能设计

* 用户管理：使用Spring Security进行身份验证，采用JWT进行用户会话管理。
* 药品管理：实现药品的CRUD操作。
* 销售管理：记录销售交易，生成销售报表。
* 库存管理：管理药品库存，提供库存预警功能。
* 供应商管理：管理供应商信息和采购订单。
* 报告与分析：生成各种报表，提供数据分析功能。
* 系统设置：配置系统参数，管理数据备份与恢复。

#### 4.3 API接口设计

* 用户管理
  + POST /api/users/login: 用户登录
  + POST /api/users/register: 用户注册
  + GET /api/users/profile: 获取用户信息
* 药品管理
  + GET /api/medicines: 获取药品列表
  + POST /api/medicines: 创建药品
  + GET /api/medicines/{id}: 获取药品详情
  + PUT /api/medicines/{id}: 更新药品
  + DELETE /api/medicines/{id}: 删除药品
* 销售管理
  + GET /api/sales: 获取销售记录
  + POST /api/sales: 记录销售交易
  + GET /api/sales/{id}: 获取销售详情
* 库存管理
  + GET /api/inventory: 获取库存信息
  + POST /api/inventory: 添加库存
  + PUT /api/inventory/{id}: 更新库存
  + DELETE /api/inventory/{id}: 删除库存
* 供应商管理
  + GET /api/suppliers: 获取供应商列表
  + POST /api/suppliers: 创建供应商
  + GET /api/suppliers/{id}: 获取供应商详情
  + PUT /api/suppliers/{id}: 更新供应商
  + DELETE /api/suppliers/{id}: 删除供应商
* 报告与分析
  + GET /api/reports/sales: 获取销售报表
  + GET /api/reports/inventory: 获取库存报表
  + GET /api/reports/financial: 获取财务报表

### 5. 前端设计

#### 5.1 项目结构

前端项目结构如下：

pharmacy-management-frontend/  
├── public/  
│ ├── index.html  
├── src/  
│ ├── assets/  
│ ├── components/  
│ ├── views/  
│ ├── router/  
│ ├── store/  
│ ├── App.vue  
│ ├── main.js  
├── package.json  
└── README.md

#### 5.2 核心功能设计

* 用户界面：提供用户登录、注册及信息管理功能。
* 药品管理界面：展示药品列表，支持药品的CRUD操作。
* 销售管理界面：记录销售交易，查看销售记录。
* 库存管理界面：管理药品库存，查看库存状态。
* 供应商管理界面：管理供应商信息，查看采购订单。
* 报告与分析界面：展示销售、库存及财务报表。
* 系统设置界面：配置系统参数，管理数据备份与恢复。

#### 5.3 页面设计

* 登录页面（Login.vue）：用户登录入口。
* 注册页面（Register.vue）：用户注册入口。
* 药品列表页面（MedicineList.vue）：展示药品列表，支持药品的CRUD操作。
* 药品详情页面（MedicineDetail.vue）：展示药品详细信息及操作入口。
* 销售记录页面（Sales.vue）：记录销售交易，查看销售记录。
* 库存管理页面（Inventory.vue）：管理药品库存，查看库存状态。
* 供应商管理页面（Supplier.vue）：管理供应商信息，查看采购订单。
* 报告页面（Report.vue）：展示销售、库存及财务报表。
* 系统设置页面（Settings.vue）：配置系统参数，管理数据备份与恢复。

### 6. 安全设计

#### 6.1 身份验证与授权

* 使用Spring Security进行用户身份验证和授权管理。
* 采用JWT（JSON Web Token）进行用户会话管理。

#### 6.2 数据加密

* 用户密码在存储前使用BCrypt进行加密。
* 重要数据传输使用HTTPS进行加密。

#### 6.3 日志与监控

* 系统操作日志记录用户重要操作行为。
* 异常日志记录系统错误和异常信息。
* 使用监控工具（如Prometheus和Grafana）监控系统性能和健康状态。

### 7. 部署设计

#### 7.1 部署环境

* 操作系统：Linux
* 应用服务器：Tomcat
* 数据库服务器：MySQL
* 前端服务器：Nginx

#### 7.2 部署流程

* 克隆项目代码库。
* 使用Maven构建后端项目并打包为WAR文件，部署到Tomcat服务器。
* 使用npm安装前端项目依赖，构建前端项目并部署到Nginx服务器。
* 配置数据库连接，初始化数据库。
* 配置Nginx代理，实现前后端分离部署。
* 启动Tomcat和Nginx服务，访问系统。